(translation of the front page of the priority document of Japanese Patent Application No. 2000-391377)

PATENT OFFICE
JAPANESE GOVERNMENT

This is to certify that the annexed is a true copy of the following application as filed with this Office.

Date of Application: December 22, 2000

Application Number : Patent Application 2000-391377
Applicant(s) : Canon Kabushiki Kaisha

December 21, 2001
Commissioner,
Patent Office Kouzo OIKAWA

Certification Number 2001-3110649

CFM2450 US

日 本 国 特 許 庁
JAPAN PATENT OFFICE

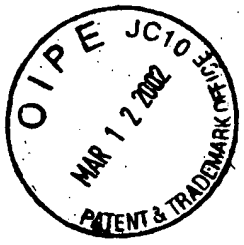
別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年１２月２２日

出願番号
Application Number: 特願２０００－３９１３７７

出願人
Applicant(s): キヤノン株式会社



CERTIFIED COPY OF PRIORITY DOCUMENT

２００１年１２月２１日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造

【書類名】　特許願
【整理番号】　4266020
【提出日】　平成12年12月22日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 15/00
【発明の名称】　情報処理装置及び情報処理方法、画像形成装置、記憶媒体
【請求項の数】　14
【発明者】
【住所又は居所】　東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内
【氏名】　川本　浩一
【特許出願人】
【識別番号】　000001007
【氏名又は名称】　キヤノン株式会社
【代理人】
【識別番号】　100076428
【弁理士】
【氏名又は名称】　大塚　康徳
【電話番号】　03-5276-3241
【選任した代理人】
【識別番号】　100101306
【弁理士】
【氏名又は名称】　丸山　幸雄
【電話番号】　03-5276-3241
【選任した代理人】
【識別番号】　100115071
【弁理士】
【氏名又は名称】　大塚　康弘

【電話番号】　03-5276-3241

【手数料の表示】

【予納台帳番号】　003458

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0001010

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　情報処理装置及び情報処理方法、画像形成装置、記憶媒体

【特許請求の範囲】

【請求項１】　設定条件を相互に整合させるための基本処理をする手段と、

前記設定条件を整合させるために、前記基本処理を補完をするための補完処理ルールを生成する生成手段と、

前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御手段と、

を備えることを特徴とする情報処理装置。

【請求項２】　前記制御手段は、設定条件を入力するための入力手段から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする請求項１に記載の情報処理装置。

【請求項３】　設定条件を可視化するためのインタフェース手段と、

前記制御手段により決定された条件を前記インタフェース手段に表示する表示制御手段と、

を備えることを特徴とする請求項１または２に記載の情報処理装置。

【請求項４】　前記表示制御手段は、前記制御手段による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする請求項３に記載の情報処理装置。

【請求項５】　請求項１乃至４のいずれかに記載の情報処理装置と、

前記情報処理装置に入力された画像を形成するための制御パラメータを決定し、該決定された制御パラメータに基づき画像情報を形成するための画像形成手段と、

を備えることを特徴とする画像形成装置。

【請求項６】　前記画像形成装置にはプリンタ、ファクシミリが含まれることを特徴とする請求項５に記載の画像形成装置。

【請求項７】　設定条件を相互に整合させるための基本処理を実行する工程と、

前記設定条件を整合させるために、前記基本処理を補完をするための補完処理ルールを生成する生成工程と、

前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御工程と、

を備えることを特徴とする情報処理方法。

【請求項８】　前記制御工程は、設定条件を入力処理するための入力工程から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする請求項７に記載の情報処理方法。

【請求項９】　設定条件を可視化させるためのインタフェース工程と、

前記制御工程により決定された条件を前記インタフェース工程で表示処理させる表示制御工程と、

を備えることを特徴とする請求項７または８に記載の情報処理方法。

【請求項１０】　前記表示制御工程は、前記制御工程による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする請求項９に記載の情報処理方法。

【請求項１１】　情報処理方法をコンピュータで実行するためのプログラムモジュールを格納したコンピュータ可読の記憶媒であって、該プログラムモジュールが、

設定条件を相互に整合させるための基本処理を実行するモジュールと、

前記設定条件を整合させるために、前記基本処理を補完をするための補完処理ルールを生成する生成モジュールと、

前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御モジュールと、

を備えることを特徴とする記憶媒体。

【請求項１２】　前記制御モジュールは、設定条件を入力処理するための入力モジュールで入力処理された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする請求項１１に記載の記憶媒体。

【請求項１３】　設定条件を可視化処理するためのインタフェースモジュールと、

前記制御モジュールにより決定された条件を前記インタフェースモジュールで表示処理させる表示制御モジュールと、

を備えることを特徴とする請求項１１または１２に記載の記憶媒体。

【請求項１４】　前記表示制御モジュールは、前記制御モジュールによる前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする請求項１３に記載の記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、情報処理装置の処理条件の設定において、複数の設定項目間で条件の不整合が生じた場合に、その不整合を解消する装置及び情報処理方法及びその方法をコンピュータで実行するためのプログラムモジュールを格納した記憶媒体、その情報処理装置を組込んだ画像形成装置に関する。

【０００２】

【従来の技術】

従来、コンピュータ上で動作するプログラムはユーザインタフェースを介してユーザから複数の動作内容を定義するための設定値の入力を受け付けると、それらの設定値に基づいて処理を実行する。

【０００３】

ところが、複数の設定条件において排他的な関係や、設定条件相互間で依存する関係等が存在する場合はユーザからの設定値の入力を受け付ける度に、入力された設定条件、設定値との関係を評価し、それらの設定条件等に不整合がないかどうかの判別を行い、不整合があった場合にはその解消を実施する処理を行なっていた。その処理の実現にあたっては、不整合の検知および不整合の解消処理を設定値間の関係に依存した条件の下で専用の処理プラグラムを用いるのが一般的である。

【０００４】

あるいは、不整合の処理が必要となる複数の設定値の条件を一覧としてまとめてファイル等に保存しておき、このファイルを不整合を解消する汎用的な処理プログラムを利用して、このプログラムに読み込ませて、設定条件の不整合を解消させる場合もある。

【０００５】

【発明が解決しようとする課題】

しかしながら、上記の不整合（以下、「コンフリクト」）をプログラムにより解消するための条件の記述は、プログラム開発者などが全ての条件を網羅的に行うため、設定値間の依存関係が複雑な場合には、全ての条件を網羅することができずに、漏れ、すなわち、不整合の状態が完全に解消されない場合が生じることがあった。

【０００６】

【課題を解決するための手段】

本発明は上記従来例における課題に鑑みてなされたもので、プログラム開発者などが用意するコンフリクトを解消するための条件を基に、この条件を補完する条件を自動生成し、コンフリクト条件に加えることで、漏れがなく、より確かなコンフリクトの解消処理を実現することを目的とする。

【０００７】

上記目的を達成するために、本発明にかかる情報処理装置、方法およびその方法をコンピュータで実行するためのプログラムモジュールを格納した記憶媒体は主として以下の構成を有することを特徴とする。

【０００８】

すなわち、情報処理装置は、設定条件を相互に整合させるための基本処理をする手段と、

前記設定条件を整合させるために、前記基本処理を補完をするための補完処理ルールを生成する生成手段と、

前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御手段と、

を備えることを特徴とする。

【０００９】

上記の情報処理装置において、前記制御手段は、設定条件を入力するための入力手段から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする。

【００１０】

上記の情報処理装置において、設定条件を可視化するためのインタフェース手段と、

前記制御手段により決定された条件を前記インタフェース手段に表示する表示制御手段と、

を備えることを特徴とする。

【００１１】

上記の情報処理装置において、前記表示制御手段は、前記制御手段による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする。

【００１２】

また、画像形成装置のおいて、上記の情報処理装置と、

前記情報処理装置に入力された画像を形成するための制御パラメータを決定し、該決定された制御パラメータに基づき画像情報を形成するための画像形成手段とを備えることを特徴とする。

【００１３】

また、情報処理方法は、設定条件を相互に整合させるための基本処理を実行する工程と、

前記設定条件を整合させるために、前記基本処理を補完をするための補完処理ルールを生成する生成工程と、

前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御工程と、

を備えることを特徴とする。

【００１４】

上記情報処理方法において、前記制御工程は、設定条件を入力処理するための入力工程から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする。

【００１５】

上記情報処理方法において、設定条件を可視化させるためのインタフェース工程と、

前記制御工程により決定された条件を前記インタフェース工程で表示処理させる表示制御工程と、

を備えることを特徴とする。

【００１６】

上記の情報処理方法において、前記表示制御工程は、前記制御工程による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする。

【００１７】

また、情報処理方法をコンピュータで実行するためのプログラムモジュールを格納したコンピュータ可読の記憶媒であって、該プログラムモジュールが、

設定条件を相互に整合させるための基本処理を実行するモジュールと、

前記設定条件を整合させるために、前記基本処理を補完をするための補完処理ルールを生成する生成モジュールと、

前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御モジュールと、

を備えることを特徴とする。

【００１８】

上記の記憶媒体において、前記制御モジュールは、設定条件を入力処理するための入力モジュールで入力処理された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする。

【００１９】

上記の記憶媒体において、設定条件を可視化処理するためのインタフェースモジュールと、

前記制御モジュールにより決定された条件を前記インタフェースモジュールで表示処理させる表示制御モジュールと、

を備えることを特徴とする。

【００２０】

上記の記憶媒体において、前記表示制御モジュールは、前記制御モジュールによる前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする。

【００２１】

【発明の実施形態】

以下、本発明を適用するのに好適である実施形態について説明を行う。

【００２２】

図１は本発明が適用される実施形態において、設定された条件が整合していないとき、この不整合を解消するための情報処理（コンフリクト解消処理）が実行される印刷処理システムの構成を説明するブロック図である。

【００２３】

なお、特に断らない限り、本発明にかかる情報処理が実行されるのであれば、単体の機器であっても、複数の機器からなるシステムであっても、ＬＡＮ,ＷＡＮ等のネットワークを介して接続が為され処理が行われるシステムであっても本発明を適用できることは言うまでもない。

【００２４】

図１において、３０００はホストコンピュータで、ＲＯＭ３ｂのプログラム用ＲＯＭあるいは外部メモリ１１に記憶された文書処理プログラム等に基づいて図形、イメージ、文字、表（表計算等を含む）等が混在した文書処理を実行するＣＰＵ１を備え、システムバス４に接続される各デバイスをＣＰＵ１が総括的に制御する。

【００２５】

また、プログラム用ＲＯＭ３ｂあるいは外部メモリ１１には、ＣＰＵ１の制御

プログラムであるオペレーティングシステムプログラム（以下ＯＳ）等を記憶し、フォント用ＲＯＭ３ａあるいは外部メモリ１１には上記の文書処理の際に使用するフォントデータ等が記憶され、データ用ＲＯＭ３ｃあるいは外部メモリ１１には文書処理等を行う際に使用する各種データが記憶されている。

【００２６】

２はＲＡＭで、ＣＰＵ１の主メモリ、ワークエリア等として機能する。５はキーボードコントローラ（ＫＢＣ）で、キーボード９や不図示のポインティングデバイスからのキー入力を制御する。６はＣＲＴコントローラ（ＣＲＴＣ）で、ＣＲＴディスプレイ（ＣＲＴ）１０の表示を制御する。

【００２７】

７はディスクコントローラ（ＤＫＣ）で、ブートプログラム、各種のアプリケーション、フォントデータ、ユーザファイル、編集ファイル、プリンタ制御コマンド生成プログラム（以下プリンタドライバ）等を記憶するハードディスク（ＨＤ）及びフロッピーディスク（ＦＤ）等の外部メモリ１１とのアクセスを制御する。

【００２８】

８はプリンタコントローラ（ＰＲＴＣ）で、所定の双方向性インターフェース（インターフェース）２１を介してプリンタ１５００に接続されて、プリンタ１５００との通信制御処理を実行する。なお、ＣＰＵ１は、例えばＲＡＭ２上に設定された表示情報ＲＡＭへのアウトラインフォントの展開（ラスタライズ）処理を実行し、ＣＲＴ１０上でのＷＹＳＩＷＹＧを可能としている。

【００２９】

また、ＣＰＵ１は、ＣＲＴ１０上の不図示のマウスカーソル等で指示されたコマンドに基づいて登録された種々のウインドウを開き、種々のデータ処理を実行する。ユーザは印刷を実行する際、印刷の設定に関するウインドウを開き、プリンタの設定や、印刷モードの選択を含むプリンタドライバに対する印刷処理方法の設定を行える。

【００３０】

プリンタ１５００において、１２はプリンタＣＰＵで、プログラム用ＲＯＭ１

３ｂに記憶された制御プログラム等あるいは外部メモリ１４に記憶された制御プログラム等に基づいてシステムバス１５に接続される印刷部（プリンタエンジン）１７に出力情報としての画像信号を出力する。また、プログラムＲＯＭ１３ｂには、ＣＰＵ１２の制御プログラム等を記憶する。フォント用ＲＯＭ１３ａには出力情報を生成する際に使用するフォントデータ等を記憶し、データ用ＲＯＭ１３ｃにはハードディスク等の外部メモリ１４がないプリンタの場合には、ホストコンピュータ上で利用される情報等を記憶している。

【００３１】

ＣＰＵ１２は入力部１８を介してホストコンピュータとの通信処理が可能となっており、プリンタ内の情報等をホストコンピュータ３０００に通知可能に構成されている。１９はＣＰＵ１２の主メモリ、ワークエリア等として機能するＲＡＭで、図示しない増設ポートに接続されるオプションＲＡＭによりメモリ容量を拡張することができるように構成されている。

【００３２】

なお、ＲＡＭ１９は、出力情報展開領域、環境データ格納領域、ＮＶＲＡＭ等に用いられる。前述したハードディスク（ＨＤ）、ＩＣカード等の外部メモリ１４は、メモリコントローラ（ＭＣ）２０によりアクセスを制御される。外部メモリ１４は、オプションとして接続され、フォントデータ、エミュレーションプログラム、フォームデータ等を記憶する。

【００３３】

また、１８は前述した操作パネルで操作のためのスイッチおよびＬＥＤ表示器等が配されている。また、前述した外部メモリは１個に限らず、少なくとも１個以上備え、内蔵フォントに加えてオプションフォントカード、言語系の異なるプリンタ制御言語を解釈するプログラムを格納した外部メモリを複数接続できるように構成されていてもよい。さらに、図示しないＮＶＲＡＭを有し、操作パネル１５０１からのプリンタモード設定情報を記憶するようにしてもよい。

【００３４】

図２に示すのが、本実施形態におけるユーザインタフェース制御プログラムがホストコンピュータ３０００上のＲＡＭ２にロードされ、実行可能となった状態

のメモリマップを示している。なお、本実施形態におけるユーザインタフェース制御プログラムは印刷処理関連プログラム２０４の一部として存在している。

【００３５】

図４は、本実施形態のコンフリクト処理における各モジュールで扱うデータの関連を示す図である。

【００３６】

図５は本実施形態におけるコンフリクトを解消するための、基本となるコンフリクトルールを補完するための補完ルールを生成するための処理（以下、「補完ルールの生成処理」という。）を説明するフローチャートである。以下、図５に示すフローチャートを中心にして本実施形態を詳しく説明する。

【００３７】

本実施形態ではプリンタドライバがＣＲＴ１０に各種設定用に表示するユーザインタフェースから設定入力された設定条件に関するコンフリクト処理を例に説明する。ユーザが図１に示したキーボード９を介して、キーボードコントローラＫＢＣ５の制御によりＣＲＴ１０に表示されたユーザインタフェース画面の一例を図８に示す。プリンタドライバＵＩ（ユーザインタフェース）を開く指示をすることでコンフリクトルールの補完処理が開始する。

【００３８】

ユーザがプリンタドライバＵＩを開く指示により、プリンタドライバＵＩを開くための初期化処理が行われ、ＯＳの管理の下、ＲＡＭ２に印刷処理関連プログラム２０４がロードされる。

【００３９】

補完ルールの生成処理を図３及び図６を用いて説明する。

【００４０】

ここで、図３は本実施形態におけるユーザインタフェース制御部の内部処理の概要を示す図である。図３において、推論エンジン３０２は図６（補完ルールの生成処理前のコンフリクト処理ルール）に例示してあるように表記されたコンフリクト処理ルール３０１をコンフリクトマネージャ３０３を介してＲＡＭ２中に読み込む（ステップＳ５０１）。

【００４１】

続いて、ステップＳ５０１で読み込んだコンフリクト処理ルールを基に、２状態値（ONとOFF）を値として有するルールについて補完ルールを生成する（ステップＳ５０２）。

【００４２】

例えば、以下の（１）から（３）は２状態値の設定状態を記述した例であり、同じプリンタの機能について以下の例のように左辺に ON または OFF のどちらかだけが記述された場合を想定すると、

A(ON) <- B(ON), C(OFF).... (1)

A(ON) <- D(V1). ... (2)

B(OFF) <- E(OFF). ... (3)

（１）の場合は設定値ＢがＯＮ、設定値ＣがＯＦＦに設定された場合に、設定値Ａの設定がＯＮになることを示しており、（２）は多状態設定値である設定値Ｄの値がＶ１に設定された場合も設定値Ａの設定が（１）の場合同様、ＯＮになることを示している。また（３）は設定値ＥがＯＦＦに設定された場合に、設定値Ｂの値がＯＦＦになることを示している。

【００４３】

この場合には推論エンジン３０２が ON/OFFを 逆にするルールを以下のように自動生成する。すなわち、上記の(1),(2) に対しては（４）式のようなコンフリクト処理ルールを生成する。

【００４４】

つまり、ＡはＯＮの状態ではなく、ＯＦＦの状態値をとる。

【００４５】

A(OFF) <- not A(ON) ... (4)

同様に(3)に対しては左辺Ｂの設定を変更するコンフリクト処理ルールを生成する。つまり、ＢはＯＦＦの状態ではなく、ＯＮの状態値をとる。

【００４６】

B(ON) <- not B(OFF) ... (5)

このように、推論エンジン３０２によって自動生成された(4),(5)のルールは

処理効率上最適化されて以下の(4)',(5)'のように変形されるが、意味は全く同じとなる。つまり、Ａの設定ＯＦＦ、Ｂの設定ＯＮは真であるという内容がルールとして生成されることになる。

【００４７】

A(OFF) <- true　　... (4)'

B(ON) <- true　　... (5)'

上記の例のように逆の状態値に変更するように自動生成されたルールは論理的に制御対象（例えばプリンタやファクシミリ等）の設定条件を定義する状態値として１００％、矛盾なく機能を遂行するための論理を整合させるものである。

【００４８】

続いて、図４に示すように状態変数に生成された補完ルールを反映する（ステップ５０３）。

【００４９】

補完ルールの生成処理において、制御対象（例えばプリンタやファクシミリ等）の全ての機能設定項目についてはコンフリクトマネージャ３０３の内部に、図４に示すような状態変数リスト４０２を持っている。この状態変数の値はプリンタドライバUIで使用される内部構造体４０１の対応するメンバ（状態変数）の値と連動している。全てのプリンタ機能設定項目の状態変数の初期値はその内部構造体４０１のメンバの値が初期値となる。

【００５０】

例えば以下の設定を記述した場合、

A(ON) <- B(ON), C(OFF)

図４の状態変数リスト４０２に示すようにコンフリクト処理ルール４０３で使用するプリンタにおける機能設定項目A，B，C のそれぞれについて同名の状態変数が存在する。そして、プリンタにおける機能設定項目A，B，C に対応する プリンタドライバUI内部構造体４０１のメンバをそれぞれ int cA，int cB，int c Cとする。

【００５１】

int cA の初期値は0であり、それに対応する A の値は０に対応する状態変数0

FF となっている。従って推論エンジン３０２内のプリンタ機能設定項目A の状態値の初期値も 状態変数「OFF」が設定される。推論エンジン３０２でコンフリクトチェックの推論が行われて以下のルール、

A(ON) <- B(ON), C(OFF)、

が成立した場合、推論エンジン３０２は左辺のプリンタ機能設定項目Aの状態変数値を「ON」に変更する。

【００５２】

コンフリクトチェックの推論が終了した後、コンフリクトマネージャは変更された状態変数の値をプリンタドライバUI内部構造体４０１の対応するメンバ int cA に反映（マッピング）する。つまり int cA は上記ルールが成立したことによって「0」から 状態変数ONに対応する「1 」に変更される。

【００５３】

推論エンジン３０２は推論エンジンに組込まれている組込み関数status(a, _X) によってプリンタ機能設定項目A の状態変数値を推論エンジン３０２で使用する変数 _X に変更する。推論エンジン３０２はコンフリクト処理ルールをロードした後、そのコンフリクト処理ルール中に出現する全ての状態変数について、コンフリクトを解消するための補完ルールを自動的に生成する。

【００５４】

A(_X) <- status(A,_X)

B(_X) <- status(B,_X)

C(_X) <- status(C,_X)

これは他に適用するルールが存在しない場合には プリンタドライバUI内部構造体４０１の対応するメンバの値がそのプリンタ機能設定項目の状態値となることを意味する。

【００５５】

プリンタの機能設定項目A については、

A(ON) <- B(ON), C(OFF)

なる関係が成り立と判断されれば A の状態値はこれを受けて ON と設定される。B がもし上記の自動生成されたルール以外に依存するルールが存在しないなら

ば、

B(_X) <- status(B,_X)

として状態変数が設定される。この自動生成ルールによりB の状態変数の値 ONが_X にユニファイされ、それがプリンタ機能B の状態値となる。

【００５６】

つまり、ユーザ定義ルールが存在しないか、または存在していてもプリンタ機能について影響が及ばない場合はプリンタドライバのUIの内部構造体４０１に対応するメンバに格納されている値がそのプリンタ機能の状態値ということになる。

【００５７】

続いて、その他の初期化処理として、プリンタドライバのUIをオープンするために必要な初期化処理を行い、図８に例示するようなプリンタドライバユーザインタフェース（UI）をオープンする（ステップＳ５０４）。プリンタドライバUIがオープンされた後は、OSより送られてくるイベントの取得とその処理を繰り返す（ステップＳ５０５）。

【００５８】

ステップＳ５０５にて取得したイベントが、ユーザがプリンタドライバUI上の設定項目を変更したイベントであるかどうかの判別を行い（ステップＳ５０６）、そうでなかった場合（Ｓ５０６－Ｎｏ）には、続いてプリンタドライバUIのクローズ要求かどうかの判別を行う（ステップＳ５１２）。クローズ要求であった場合（Ｓ５１２－Ｙｅｓ）には、終了処理を行い、プリンタドライバUIをクローズして、全ての処理を終了する（ステップＳ５１３）。一方、クローズ要求でもなかった場合（Ｓ５１２－Ｎｏ）には、再び、処理をステップＳ５０５に戻す。

【００５９】

ステップＳ５０６での判別により、ステップＳ５０５で取得したイベントがユーザの設定変更要求であった場合には、ステップＳ５０１からＳ５０３の処理により構築した補完されたコンフリクト処理ルールを適用する（ステップＳ５０７）。

【００６０】

ユーザの設定変更要求が図８に示すPrint Styleを「1-Sided Printing」から「Booklet Printing」に変更するものであった場合を例にとると、コンフリクト処理ルールの適用は、図６に示すコンフリクト処理ルール３０１の一部として読み込まれた基本となるコンフリクト処理ルールと、ステップＳ５０２にて生成された補完ルールの一部である図７(４)、(5)とステップＳ５０３にて行われた状態変数を反映する図７(6)～(9)に対して行われる。

【００６１】

プリンタドライバUI内部構造体４０１のメンバとして存在するCollate、Group、Staple、Layoutの各メンバのコンフリクト処理ルール適用前の値は、以下の通りとなる。

【００６２】

Collate　　: OFF

Group　　: ON

Staple　　: OFF

Layout　　: 1-Sided。

【００６３】

ユーザの変更要求がLayoutを１－SidedからBookletに変更するものであるので、Layoutのメンバの内容は「1-Sided」から「Booklet」に変更となる。

【００６４】

Collate　　: OFF

Group　　: ON

Staple　　: OFF

Layout　　: Booklet。

【００６５】

すると、プリンタドライバUI３０６はコンフリクトマネージャ３０３を呼び出し、状態変数リストにあるLayoutの状態変数が上記のように更新され、続いて推論エンジン３０２がコールされて、コンフリクト処理ルールの適用が始まる。

【００６６】

まず、図７(6)～(9)のルールが適用され、推論エンジン３０２内の各プリンタ

機能設定項目が状態変数リストの各メンバの持つ値で初期化される。

【００６７】

続いて、図７(3)が適用され、Groupの値はONからOFFへと変更となる。

【００６８】

Collate　　：OFF

Group　　：OFF

Staple　　：OFF

Layout　　：Booklet

さらに、図７(4)のルールが適用され、CollateがOFFからONへと変更となる。

【００６９】

Collate　　：ON

Group　　：OFF

Staple　　：OFF

Layout　　：Booklet

他に、適用されるルールが存在しないので、以上で推論エンジン３０２でのコンフリクト処理ルールの適用が終了する。

【００７０】

次のステップでは、コンフリクトマネージャ３０３が上記の最終状態を基に状態変数リストの更新（ステップ５０８）とプリンタドライバUI内部構造体の更新（ステップ５０９）を行う。

【００７１】

続いて、プリンタドライバUI３０６がプリンタドライバUI内部構造体のメンバの値を参照して、UIの更新が必要かどうかの判別を行う（ステップ５１０）。

【００７２】

UIの更新の必要のない場合には、そのまま処理をステップS５０５に戻す。更新が必要な場合には、UIの更新を行い（ステップ５１１）、ステップS５０５に処理を戻す。

【００７３】

上記の例では、Layoutが「１－Sided Printing」から「Booklet Printing」に

設定が変更されたことにより、Collateが「OFF」から「ON」へ、Groupが「ON」から「OFF」へと変化しているので、プリンタドライバUIの表示も図９に示すとおりに更新される。

【００７４】

以上の処理は、プリンタドライバUIがクローズされるまで、繰り返し実行される。プリンタドライバUIがクローズされると処理は全て終了し、本実施形態における印刷処理関連プログラムの処理も終了し、RAM２からはＯＳ４０５の機能により消去される。なお、本実施形態においては、本印刷処理関連プログラムを記録する媒体を外部メモリとしているが、外部メモリとしては、FD、HDドライブ、CD-ROMやICメモリカード等であってもよい。更に、本印刷プログラム単独、もしくはOSその他のホストコンピュータ上で動作するプログラムと共にROM３ｂに記録しておき、これをメモリマップの一部となすように構成し、直接CPU１で実行することも可能である。

【００７５】

以上説明したように、本実施形態によれば、プログラム開発者が用意する基本となるコンフリクト処理ルールを補完するコンフリクト条件を自動生成することで、漏れのない、コンフリクト解消処理が実現できるという効果がある。

【００７６】

【他の実施形態】

上記実施形態においては、プリンタドライバUIの更新処理をプリンタドライバUI本体３０６で実施しているが、図１０に示すように、コンフリクト処理ルールの中に、プリンタドライバUIを更新するための処理を関数として記述し、推論エンジン３０２がその記述を解釈した時点で、コンフリクトマネージャ３０３の状態変数リスト３０４を介して、プリンタドライバのUI更新処理を直接行うようにしても良い。

【００７７】

図１０は、図７のコンフリクト処理ルールのうち、(3)にUI更新の処理を追加したものである。{disable}の記述を追加することにより、図９（プリンタドライバの表示するユーザインタフェースの例で、図８の状態で、ユーザが[Booklet

Printing]を選択した状態を例示する図）にあるGroupラジオボタンコントロールをdisableする処理がコンフリクト処理ルールの記述の一部として実現されるようになる。

【００７８】

さらに、図１１に示すようにコンフリクト処理ルールの中に、ユーザに対する情報の表示を可能にするメッセージボックスの表示処理を加えることも可能である。

【００７９】

例えば、図１１の(3)の｛ Message(MSG001) ｝の記述は、図１２のメッセージボックスを表示するための処理を示している。MSG001は図１２に表示されているメッセージテキスト「Groupの設定はCollateに調整されました。」の文字列を指し示すIDで、ID: MSG001とそのIDが示す文字列はコンフリクトマネージャ３０３に文字列リソースとして存在している。

【００８０】

本実施形態によれば、ユーザインタフェースの更新処理やメッセージ処理もコンフリクト処理ルールに加えることで、開発者にとって可読性の高い、メンテナンスの容易なコーディングを実現できるという効果がある。

【００８１】

更に、ユーザインタフェース制御部は、ユーザインタフェース更新処理とメッセージ表示をコンフリクト処理部分に設けることで、コンフリクト処理ルールが変更された場合でも容易に変更後の条件に適合した表示を実現することが可能にする。

【００８２】

【発明の効果】

以上説明したように、本発明によれば、基本となるコンフリクト処理ルールを補完する条件を自動生成するので、漏れがなくコンフリクト状態を解消する処理が実現できるという効果がある。

【００８３】

さらに他の発明によれば、ユーザインタフェースの更新処理、メッセージ処理

もコンフリクト処理ルールに加えることで、開発者にとって可読性の高い、メンテナンスの容易なコーディングを実現できるという効果がある。

【００８４】

更に、ユーザインタフェース制御部は、ユーザインタフェース更新処理とメッセージ表示をコンフリクト処理部分に設けることで、コンフリクト処理ルールが変更された場合でも容易に変更後の条件に適合した表示を実現することが可能にする。

【図面の簡単な説明】

【図１】

本発明が適用される実施形態において、コンフリクト解消処理が実行される印刷処理システムの構成を説明するブロック図である。

【図２】

本実施形態におけるユーザインタフェース制御プログラムがホストコンピュータ３０００上のＲＡＭ２にロードされ、実行可能となった状態のメモリマップを示している。

【図３】

本実施形態におけるユーザインタフェース制御部の内部処理の概要を示す図である。

【図４】

本実施形態のコンフリクト処理における各モジュールで扱うデータの関連を示す図である。

【図５】

本実施形態におけるコンフリクトを解消するための、補完ルールの生成処理を説明するフローチャートである。

【図６】

補完ルールの生成処理前のコンフリクト処理ルールを例示する図である。

【図７】

本実施形態における補完後のコンフリクト処理ルールを例示する図である。

【図８】

本実施形態におけるプリンタドライバの表示するユーザインタフェースを例示する図である。

【図９】

本実施形態におけるプリンタドライバの表示するユーザインタフェースの例で、ユーザが[Booklet Printing]を選択した状態を例示する図である。

【図１０】

他の実施形態におけるコンフリクト処理ルールの記述を例示した図である。

【図１１】

他の実施形態におけるコンフリクト処理ルールの記述を例示した図である。

【図１２】

他の実施形態におけるユーザに表示されるメッセージボックスを例示する図である。

【符号の説明】

1 CPU

2 RAM

3 ROM

4 システムバス

12 CPU

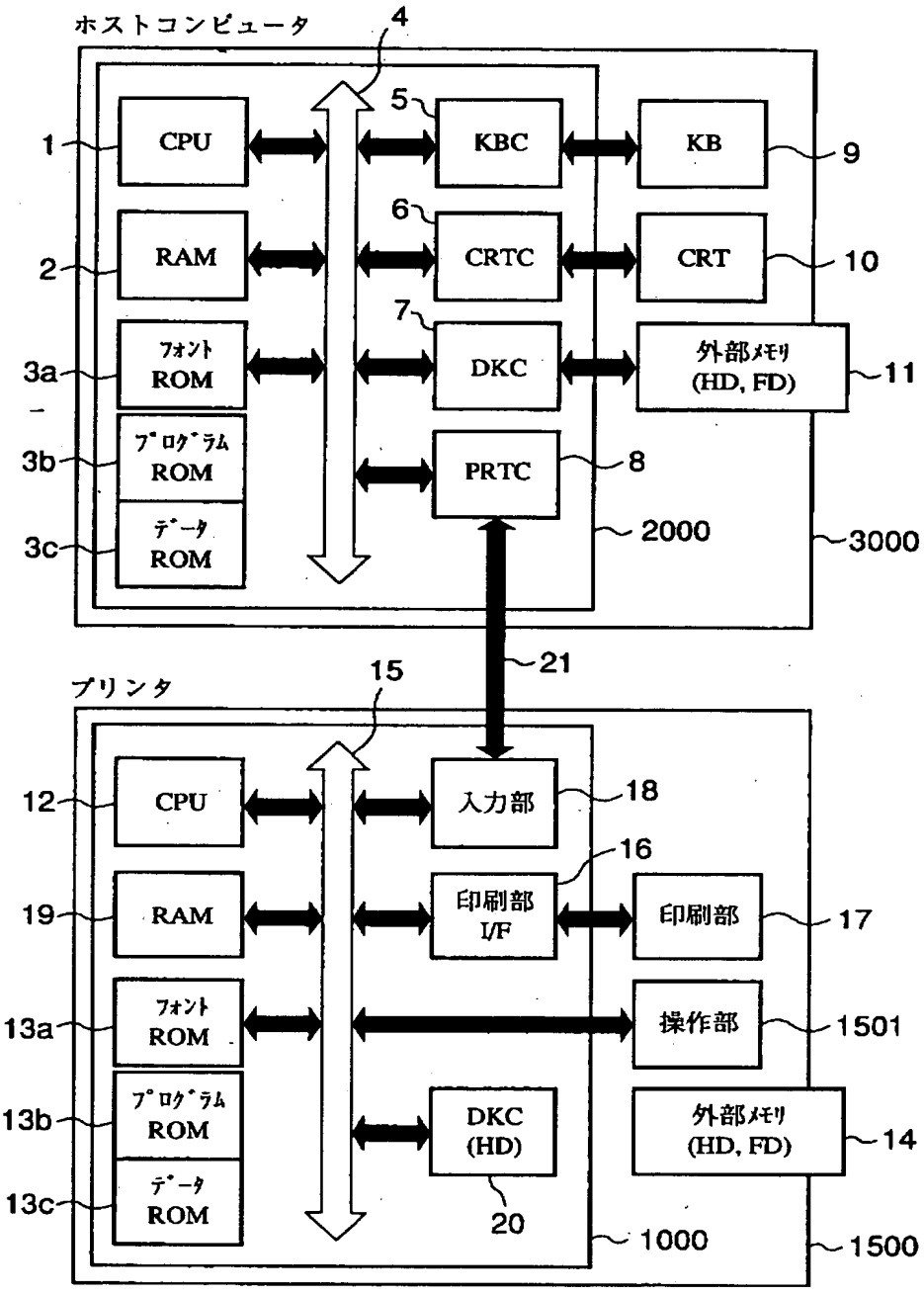
13 ROM

19 RAM

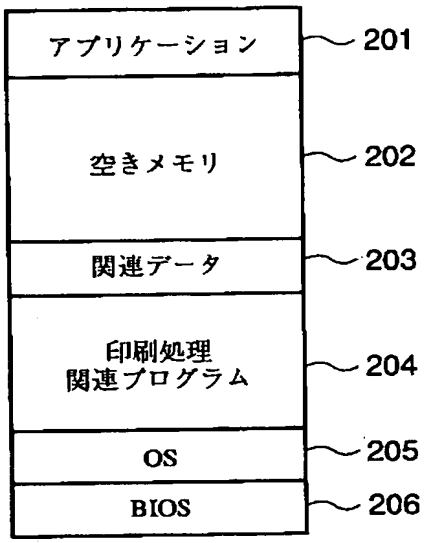
3000 ホストコンピュータ

【書類名】　　図面

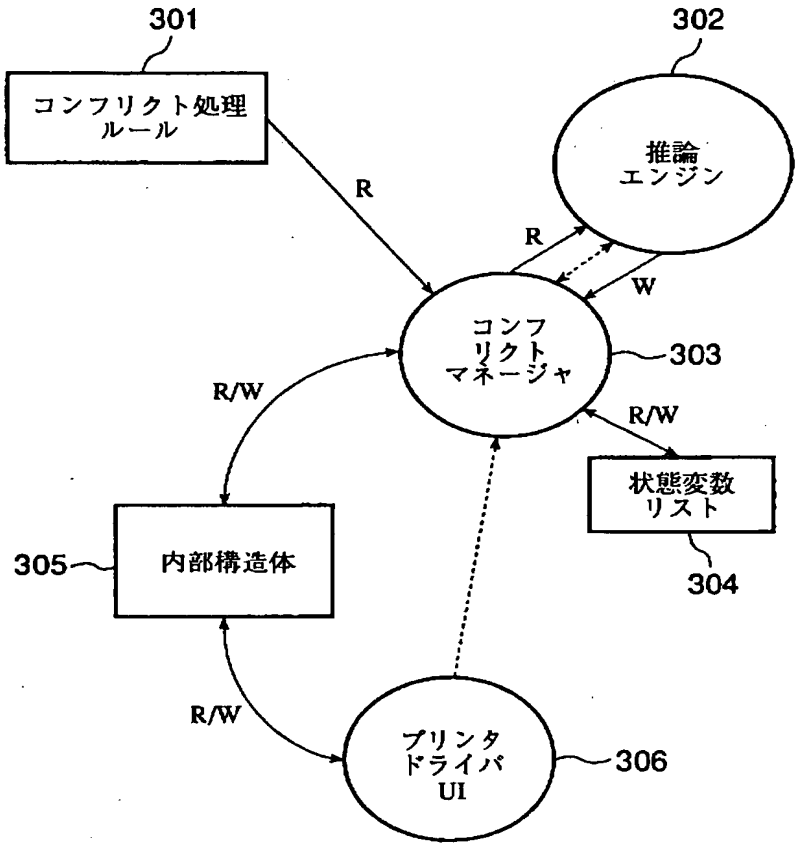
【図１】


ホストコンピュータ
4
5
KBC
KB
9
1
CPU
6
CRTC
CRT
10
2
RAM
7
DKC
外部ﾒﾓﾘ
(HD, FD)
11
3a
ﾌｫﾝﾄ
ROM
3b
ﾌﾟﾛｸﾞﾗﾑ
ROM
PRTC
8
3c
ﾃﾞｰﾀ
ROM
2000
3000
21
15
プリンタ
12
CPU
入力部
18
16
19
RAM
印刷部
I/F
印刷部
17
13a
ﾌｫﾝﾄ
ROM
操作部
1501
13b
ﾌﾟﾛｸﾞﾗﾑ
ROM
DKC
(HD)
外部ﾒﾓﾘ
(HD, FD)
14
13c
ﾃﾞｰﾀ
ROM
20
1000
1500

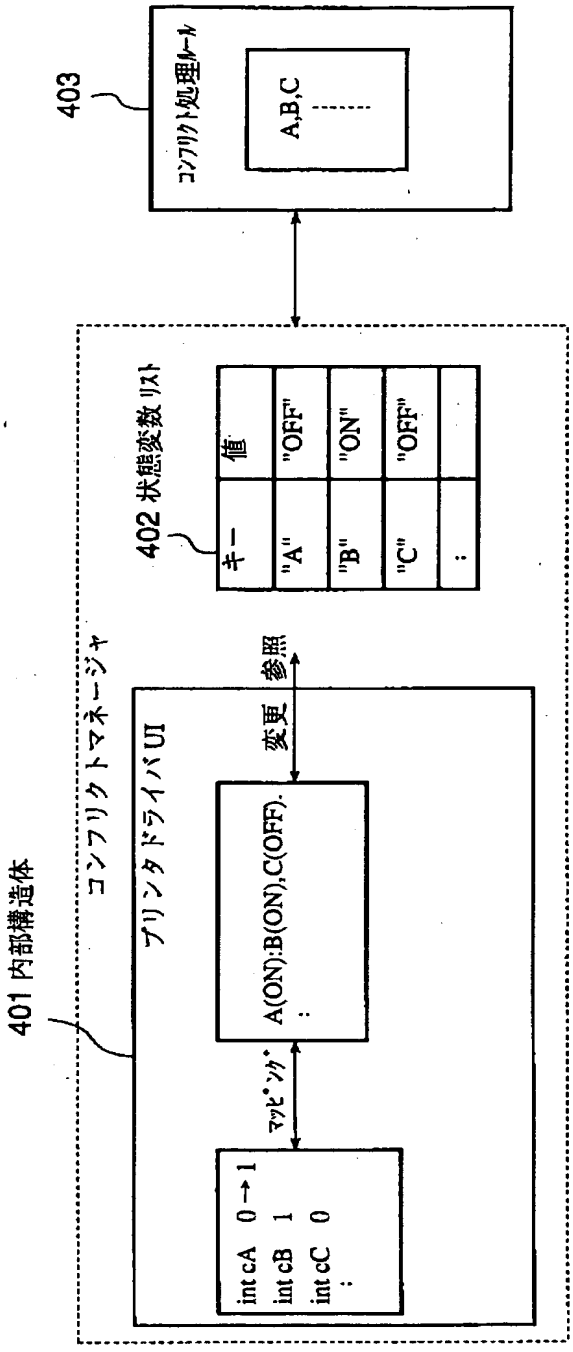
【図２】


アプリケーション
201
空きメモリ
202
関連データ
203
印刷処理
関連プログラム
204
OS
205
BIOS
206

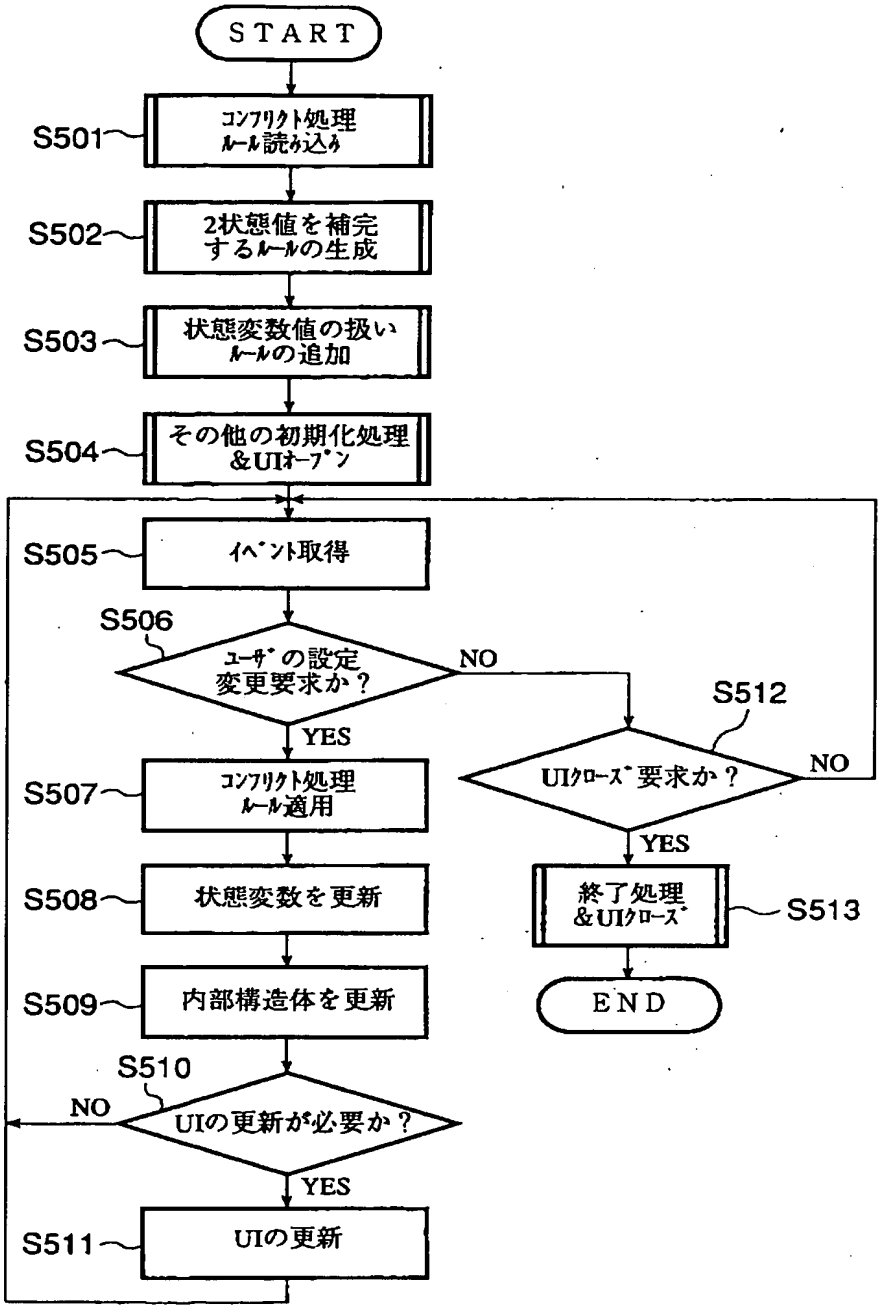
【図３】


301
コンフリクト処理
ルール
302
推論
エンジン
R
R
W
コンフ
リクト
マネージャ
303
R/W
R/W
状態変数
リスト
304
305
内部構造体
R/W
プリンタ
ドライバ
UI
306

【図４】


403
コンフリクト処理ルール
A,B,C
402 状態変数 リスト
キー
値
"A"
"OFF"
"B"
"ON"
"C"
"OFF"
401 内部構造体
コンフリクトマネージャ
プリンタドライバ UI
参照
変更
A(ON):B(ON),C(OFF).
マッピング
int cA 0→1
int cB 1
int cC 0

【図５】


START
S501
コンフリクト処理
ルール読み込み
S502
2状態値を補完
するルールの生成
S503
状態変数値の扱い
ルールの追加
S504
その他の初期化処理
&UIオープン
S505
イベント取得
S506
ユーザの設定
変更要求か？
NO
YES
S507
コンフリクト処理
ルール適用
S508
状態変数を更新
S509
内部構造体を更新
S510
UIの更新が必要か？
NO
YES
S511
UIの更新
S512
UIクローズ要求か？
NO
YES
終了処理
&UIクローズ
S513
END

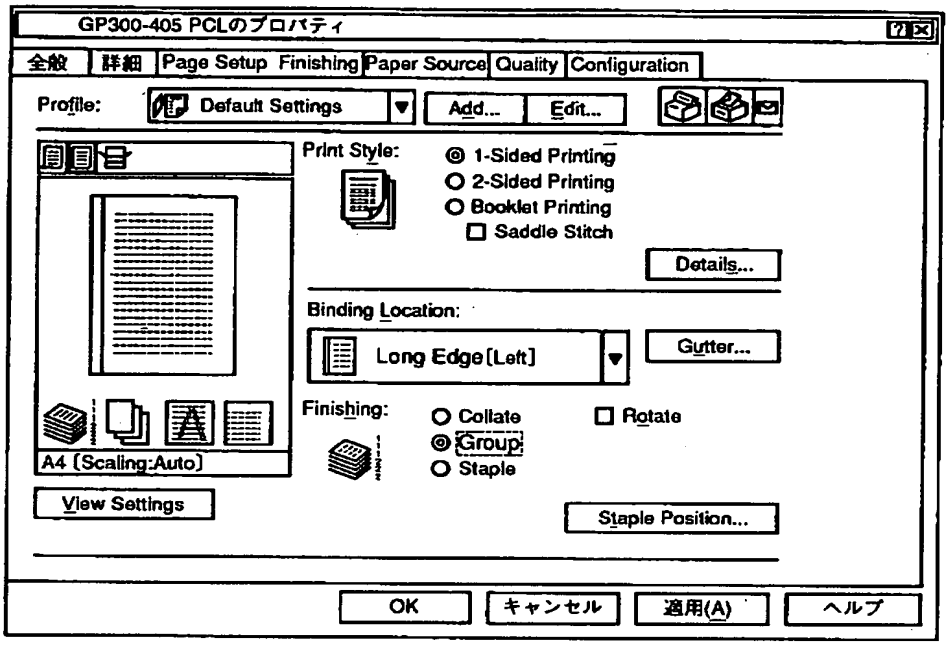
【図６】

```
Collate(0FF)<- Group(ON).          (1)
Collate(0FF)<- Staple(ON).         (2)
Group(0FF)<- Layout(BOOKLET).      (3)
```

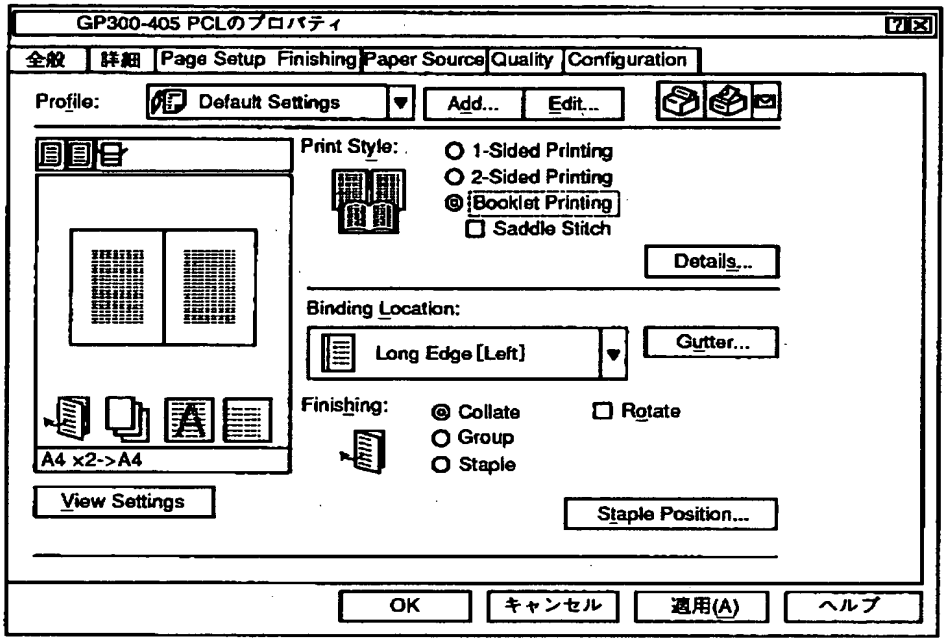
【図７】

```
Collate(0FF)<- Group(ON).          (1)
Collate(0FF)<- Staple(ON).         (2)
Group(0FF)<- Layout(BOOKLET).      (3)

Collate(0N)<- true.                (4)
Group(0N)<- true.                  (5)

Collate(_X)<- status(Collate,_X).  (6)
Group(_X)<- status(Group,_X).      (7)
Layout(_X)<- status(Layout,_X).    (8)
Staple(_X)<- status(Staple,_X).    (9)
```

【図８】


GP300-405 PCLのプロパティ
全般
詳細
Page Setup
Finishing
Paper Source
Quality
Configuration
Profile:
Default Settings
Add...
Edit...
Print Style:
1-Sided Printing
2-Sided Printing
Booklet Printing
Saddle Stitch
Details...
Binding Location:
Long Edge (Left)
Gutter...
Finishing:
Collate
Rotate
Group
Staple
A4 (Scaling:Auto)
View Settings
Staple Position...
OK
キャンセル
適用(A)
ヘルプ

【図９】


GP300-405 PCLのプロパティ
全般
詳細
Page Setup
Finishing
Paper Source
Quality
Configuration
Profile:
Default Settings
Add...
Edit...
Print Style:
1-Sided Printing
2-Sided Printing
Booklet Printing
Saddle Stitch
Details...
Binding Location:
Long Edge [Left]
Gutter...
Finishing:
Collate
Group
Staple
Rotate
A4 x2->A4
View Settings
Staple Position...
OK
キャンセル
適用(A)
ヘルプ

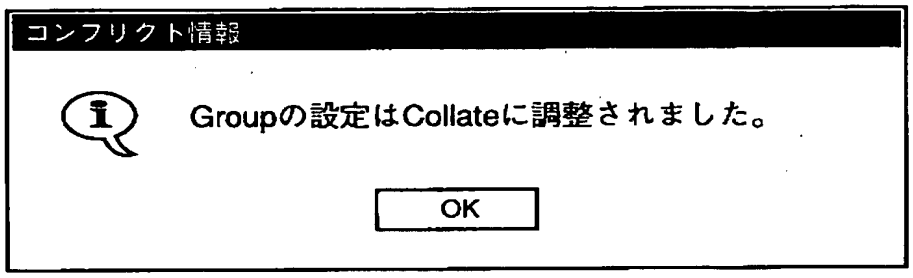
【図１０】

```
Collate(0FF)<- Group(ON).                    (1)
Collate(0FF)<- Staple(ON).                   (2)
Group(0FF)<- Layout(BOOKLET).                (3)
                {disable}
Collate(0N)<- true.                          (4)
Group(0N)<- true.                            (5)
Collate(_X)<- status(Collate,_X).            (6)
Group(_X)<- status(Group,_X).                (7)
Layout(_X)<- status(Layout,_X).              (8)
Staple(_X)<- status(Staple,_X).              (9)
```

【図１１】

```
Collate(0FF)<- Group(ON).                    (1)
Collate(0FF)<- Staple(ON).                   (2)
Group(0FF)<- Layout(BOOKLET).                (3)
                {disable}
                {Message(MSG001)}
Collate(0N)<- true.                          (4)
Group(0N)<- true.                            (5)
Collate(_X)<- status(Collate,_X).            (6)
Group(_X)<- status(Group,_X).                (7)
Layout(_X)<- status(Layout,_X).              (8)
Staple(_X)<- status(Staple,_X).              (9)
```

【図１２】



【書類名】　要約書

【要約】

【課題】　基本コンフリクト処理ルールを補完するルールを生成して、漏れなくコンフリクト状態を解消する。

【解決手段】　設定条件を相互に整合させるための基本処理を実行し、設定条件を整合させるために、その基本処理を補完をするための補完処理ルールを生成し（Ｓ５０１、Ｓ５０２）、その基本処理及び補完処理ルールに従い、設定条件を整合させて、その条件に基づく制御パラメータを決定する（Ｓ５０７，Ｓ５０８）。

【選択図】　図５

出　願　人　履　歴　情　報

識別番号　　〔０００００１００７〕

１．変更年月日　　１９９０年　８月３０日
［変更理由］　新規登録
住　所　東京都大田区下丸子３丁目３０番２号
氏　名　キヤノン株式会社